Finite State Machine과 Model checker를 사용한 Flutter 기반의 GUI 어플리케이션 UI 테스팅 자동화

202173154 송승현

Flutter Framework

2018년 구글에서 발표한 오픈소스 크로스 플랫폼 GUI 어플리케이션 프레임워크이다. Flutter Engine을 사용하여 서로 다른 운영체제 플랫폼에서 하나의 소스코드로 GUI 어플리케이션을 작성할 수 있으며, Widget이라는 UI Component로 GUI를 구성한다는 특징이 있다.

Widget의 종류

Widget은 크게 Stateless widget과 Stateful widget으로 구분되며, Stateless widget은 한번 빌드 후 화면을 다시 빌드하지 않는 경우 사용되며, Stateful widget의 경우 화면이 최초 빌드 된 이후 state(상태)의 변화에 따라 화면을 다시 빌드하는 경우 사용된다.

Widget의 Context

Widget은 Widget tree를 통해 Flutter app 내에서 관리된다. 그림1과 같이 root widget에서부터 화면에 그리기 시작하여 tree 가장 아래에 있는 Widget까지 순차적으로 화면에 빌드 된다. 이때 모든 Widget은 context 값을 가지게 되는데 이 context는 오직 한 Widget 당 하나의 context만 존재하여 Widget Tree내의 Widget의 위치를 파악하는데 사용된다. 예를 들어 그림 1 Widget tree의 최하단의 Text widget이 빌드 될 경우, 그 부모 Widget으로부터 너비와 높이 값을 받아올 수 있다. 이때 Flutter의 build 시스템은 Text widget이 가지고 있는 부모 widget의 context값을 통해 부모 widget이 widget tree내에서 어디에 있는지 파악하고 부모 widget으로부터 필요한 값들을 받아와서 자식 위젯을 build 하는데 사용한다.

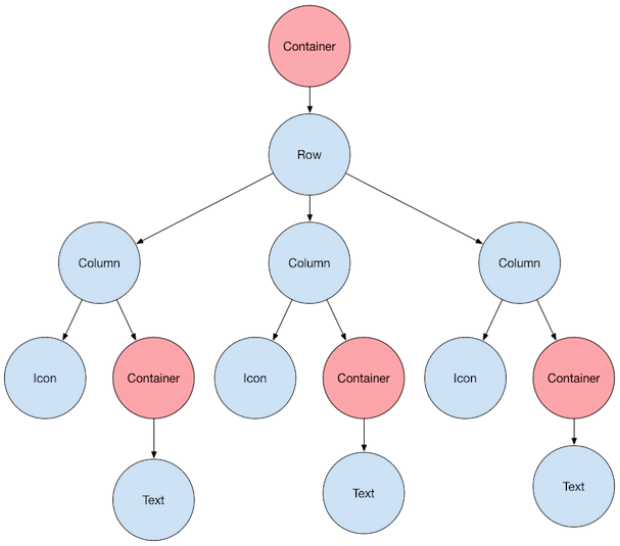

그림1 Widget tree of Flutter

Stateful Widget의 build 과정

Stateless widget과 달리 Stateful widget은 사용자의 조작 혹은 데이터의 변화에 따라 화면을 다시 빌드하거나 다른 Widget을 반환하는 과정을 거친다. 그림2는 Stateful Widget의 생명주기를 설명하고 있다.

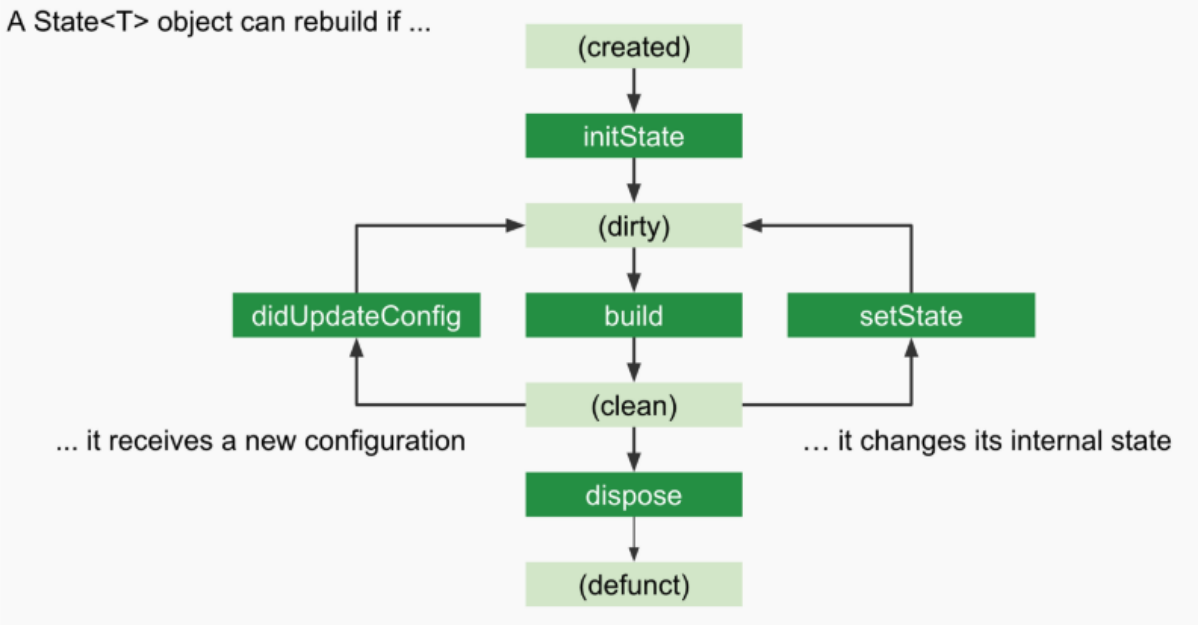

그림 2 Life Cycle of Stateful widget

최초로 widget이 만들어질 경우 initState가 호출되며, 이때 context가 할당된다. 이후 state의 변화가 생길 경우 didUpdateConfig 혹은 setState를 통해 dependancy와 state의 값을 변화시킨 후 이를 widget에 반영하여 build를 호출한다. 이후 widget의 사용이 끝나면 dispose 해주며 context 할당을 해제해준다.

Flutter GUI application의 테스팅

Flutter Framework는 다음과 같은 테스팅을 제공한다.

- Unit Testing
- Widget Testing
- Integration Testing

	Unit	Widget	Integration
Confidence	Low	Higher	Highest
Maintenance cost	Low	Higher	Highest
Dependencies	Few	More	Most
Execution speed	Quick	Quick	Slow

유닛 테스트는 어플리케이션이 완성하지 않은 상태에서 작은 단위의 함수, 클래스 등을 테스트하며, Widget Test는 테스트하고자 하는 위젯이 사용자의 이벤트를 수신 받고, 하위 위젯을 인스턴스화 할 수 있는지 파악한다. Integration Test는 거의 완성된 App을 테스트하며 사용자의 행동 등을

정의할 수 있다.

Flutter 기존 테스팅 방법의 문제점

- 테스트 시나리오를 사용자가 직접 작성해 주며, state의 모든 상태에 대하여 탐색이 불가능 하다.
- 즉, 개발자가 작성한 범위 내에서 테스트가 작동하며, 개발자가 누락한 test가 존재할 수 있다.
- Test 비용과 시간이 많이 걸린다.

Finite State Machine으로 표현된 GUI application과 GUI 테스팅에 model checking 활용

GUI application에서 state management의 중요성이 증대됨에 따라 FSM(Finite State Machine)을 GUI Framework에 FSM을 state management에 활용하려는 시도가 제안되고 있다. 이번 Software model checking 수업을 진행하면서 Web GUI가 FSM으로 표현될 수 있다면, 이를 통해 "GUI의 빌드 중 빈번하게 일어나는 문제들을 Model checking을 통해 빌드 이전에 발견할 수 있을 것"이라는 가설을 세우게 되었음.

Proposal

모델링 하고자 하는 소프트웨어: 사용자의 event, 혹은 서버로부터 데이터를 수신하는 Flutter GUI 어플리케이션

Model checking을 통해 검증하고자 하는 property:

- 해상도에 따라서 overflow를 발생하지 않을 것
- 화면이동에 Navigator를 사용시 context missing 문제가 없을 것
- 각 화면이 사용이 종료될 시 widget tree로부터 할당 해제가 이루어질 것